# 02 Arrays and Strings

- Test your knowledge

1. When to use String vs. StringBuilder in C# ?

   Since string is immutable, it's good to store some sensitive data, such as username and password. String Builder is mutable, usually it is used inside the method implementation, so that we can modify the text based on the needs , and convert the string builder into string once the operation finished.

2. What is the base class for all arrays in C#?

   Object class

3. How do you sort an array in C#?

   Array.Sort() method

4. What property of an array object can be used to get the total number of elements in an array?

   Array.Length property

5. Can you store multiple data types in System.Array?

   If we create an object array, we are able to store multiple data types in an array

   https://www.c-sharpcorner.com/UploadFile/955025/C-Sharp-interview-questions-part1/

6. What's the difference between the System.Array.CopyTo() and System.Array.Clone()?

   System.Array.Clone() will return a shallow copy of the array. A shallow copy of an Array copies only the elements of the Array, whether they are reference types or value types, but it does not copy the objects that the references refer to. The references in the new Array point to the same objects that the references in the original Array point to.

   System.Array.CopyTo() will copy all the elements of the current one-dimensional array to the specified one-dimensional array. This method copies all the elements of the current array instance to the destination array, starting at index. The destination array must already have been dimensioned and must have a sufficient number of elements to accommodate the copied elements. Otherwise, the method throws an exception.

- Practice Arrays

1. Copying an array:

```
public void CopyAnArray(int[] array)
{
    int[] newArr = new int[array.Length];
    for (int i = 0; i < array.Length; i++)
    {
        newArr[i] = array[i];
        Console.WriteLine($"The {i}th element of array is {array[i]}");
        Console.WriteLine($"The {i}th element of copied array is {newArr[i]}");
        Console.WriteLine();
    }
}
```

2. Manage list of elements

```
public void ManageElements()
{
  List<string> list = new List<string>();
  Console.Write("Enter command (+ item, - item or -- to clear): ");
```

```
  string operation = Console.ReadLine();
  while (operation != null)
  {
      switch (operation)
      {
          case "+":
              Console.Write("Please enter the element: ");
              list.Add(Console.ReadLine());
              Console.WriteLine($"The current list is: ");
              foreach (var item in list)
              {
                  Console.Write(item + "\t");
              }
              Console.WriteLine();
              break;
          case "-":
              list.RemoveAt(list.Count - 1);
              Console.WriteLine($"The current list is: {list}");
              Console.WriteLine($"The current list is: ");
              foreach (var item in list)
              {
                  Console.Write(item + "\t");
              }
              Console.WriteLine();
              break;
          case "--":
              list.Clear();
              Console.WriteLine("You have removed all the elements");
              break;
          default:
              Console.WriteLine("Invalid operation");
              break;
      }
      Console.WriteLine("Insert another input to Continue: ");
      operation = Console.ReadLine();
  }
}
```

3. Find prime numbers in a given range

```
public int[] FindPrimesInRange(int startNum, int endNum)
{
    if (startNum <= 0 || endNum <= 0 || startNum > endNum)
    {
        return null;
    }
    List<int> res = new List<int>();
    for (int i = startNum; i <= endNum; i++)
    {
        if (CheckPrime(i))
        {
            res.Add(i);
        }
        else
        {
            continue;
        }
    }
    return res.ToArray();
}

private bool CheckPrime(int a)
{
    for (int i = 2; i <= Math.Sqrt(a); i++)
    {
        if (a % i == 0)
        {
            return false;
        }
    }
    return true;
```

```
}

//call FindPrimesInRange in program.cs:
DayTwoSolution solution = new DayTwoSolution(); // I implement this method inside DayTwoSolution class
int[] res = solution.FindPrimesInRange(2, 20);
foreach (var item in res)
{
    Console.WriteLine(item);
}
```

4. Get the sum after rotation

```
public int[] SumAfterRotation(int[] arr, int k)
{
  int times = k / arr.Length;
  int move = k % arr.Length;
  int sum = 0;
  int[] res = new int[arr.Length];
  foreach (var item in arr)
  {
      sum = sum + item;
  }
  sum = sum * times;
  for (int i = 0; i < arr.Length; i++)
  {
      for (int j = 1; j <= move; j++)
      {
          if (i - j >= 0)
          {
              res[i] = res[i] + arr[i - j];
          }
          else
          {
              res[i] = res[i] + arr[i - j + arr.Length];
          }
      }
      res[i] = sum + res[i];
  }
  return res;
}
```

5. Find the longest sequence of equal elements in an int array

```
public int[] LongestSequence(int[] arr)
{
    int num = arr[0];
    int maxCount = 1;
    int count = 1;
    for (int i = 1; i < arr.Length; i++)
    {
        if (arr[i] == arr[i - 1])
        {
            count++;
            if (count > maxCount)
            {
                maxCount = count;
                num = arr[i];
            }
        }
        else
        {
            count = 1;
        }
    }
    int[] res = new int[maxCount];
    Array.Fill(res, num);
    return res;
}
```

6. most frequent number

```
public void MostFrequentNumber(int[] arr)
{
    Dictionary<int, int> freq = new Dictionary<int, int>();
    Dictionary<int, int> firstOccurrence = new Dictionary<int, int>();
    int leftmost = int.MaxValue;
    for (int i = 0; i < arr.Length; i++)
    {
        if (!freq.ContainsKey(arr[i]))
        {
            freq.Add(arr[i], 1);
            firstOccurrence.Add(arr[i], i);

        }
        else
        {
            freq[arr[i]]++;
        }
    }
    int mostFreq = freq.Values.Max();
    int num = -1;
    foreach (var key in freq.Keys)
    {
        if (freq[key] == mostFreq && firstOccurrence[key] < leftmost)
        {
            leftmost = firstOccurrence[key];
            num = arr[leftmost];
        }
    }
    Console.WriteLine($"The number {num} is the most frequent (occurs {mostFreq} times)");
}
```

- Practice strings

1. Reverse a string

```
public string ReverseStringOne(string s)
{
    if (s == null)
    {
        return string.Empty;
    }
    char[] chars = s.ToCharArray();
    for (int i = 0, j = chars.Length - 1; i < j; i++, j--)
    {
        char c = chars[i];
        chars[i] = chars[j];
        chars[j] = c;
    }
    string res = new string(chars);
    return res;
}
```

2. reverse the words

```
public string ReverseWords(string s)
{
    if (s == null)
    {
        return string.Empty;
    }
    char[] separators = new char[] {'.', ',', ':', ';', '=', '(', ')', '&', '[', ']', '"', '\'', '/', '\\', '!', '?', ' '};
    char[] chars = s.ToCharArray();
    int n = chars.Length;
```

```
    bool isWord = true;
    StringBuilder sb = new StringBuilder();
    List<string> wordsList = new List<string>();
    List<string> separatorList = new List<string>();
    foreach (var c in chars)
    {
        if (isWord)
        {
            if (separators.Contains(c))
            {
                wordsList.Add(sb.ToString());
                sb.Clear();
                isWord = !isWord;
            }
        }
        else
        {
            if (!separators.Contains(c))
            {
                separatorList.Add(sb.ToString());
                sb.Clear();
                isWord = !isWord;
            }
        }
        sb.Append(c);
    }
    separatorList.Add(sb.ToString());
    sb.Clear();
    int m = wordsList.Count();
    for (int i = 0; i < m; i++)
    {
        sb.Append(wordsList[m - 1 - i]);
        sb.Append(separatorList[i]);
    }
    return sb.ToString();
}
```

3. Extract palindrome

```
public string[] FindPalindrome(string s)
{
    if (s == null)
    {
        return null;
    }
    HashSet<string> wordsSet = new HashSet<string>();
    bool isWord = true;
    StringBuilder sb = new StringBuilder();
    char[] chars = s.ToCharArray();
    foreach (var c in chars)
    {
        if (isWord)
        {
            if (!char.IsLetter(c))
            {
                wordsSet.Add(sb.ToString());
                sb.Clear();
                isWord = !isWord;
            }
            else
            {
                sb.Append(c);
            }
        }
        else
        {
            if (char.IsLetter(c))
            {
                sb.Append(c);
                isWord = !isWord;
            }
```

```
        }
    }
    foreach (var word in wordsSet)
    {
        if (!IsPalindrome(word))
        {
            wordsSet.Remove(word);
        }
    }
    return wordsSet.ToArray();
}

private bool IsPalindrome(string s)
{
    if (s.Length <= 1)
    {
        return true;
    }
    int i = 0, j = s.Length - 1;
    while (i < j)
    {
        if (s[i] != s[j])
        {
            return false;
        } else
        {
            i++;
            j--;
        }
    }
    return true;
}
```

4. Parse URL

```
public void ParseUrl(string url)
{
    char[] seperator = new char[] { ':', '/', '/' };
    int i = url.IndexOfAny(seperator);
    string protocal, secondHalf;
    if (i != -1)
    {
        protocal = url.Substring(0, i);
        secondHalf = url.Substring(i + 3);
    }
    else
    {
        protocal = " ";
        secondHalf = url;
    }
    string[] temp = secondHalf.Split('/');
    string server = temp[0];
    string resource;
    if (temp.Length > 1)
    {
        resource = temp[1];
    } else
    {
        resource = " ";
    }
    Console.WriteLine($"[protocal] = {protocal}");
    Console.WriteLine($"[server] = {server}");
    Console.WriteLine($"[resource] = {resource}");
}
```